ZETA SOFTWARE GMBH

Zeta Test

User's Manual

Version 1.0

2009-05-21

This manual functions as an overview, introduction and reference for Zeta Test Version 1.0, a test management enviornment.

Table of contents

1 Introduction

Welcome to the Zeta Test user's manual!

Zeta Test is an intuitive-to-use test management environment/software tool for performing tests, including Black Box and White Box Tests, Regression Tests (see glossary in chapter 5.14) or Change Management Tests for software release changes.

Zeta Test helps you to plan, perform, log and document tests, as well as supporting you when evaluating and interpreting results.

Create and manage your test cases and test plans with Zeta Test. Test your software with test scripts that you created with Zeta Test.

That said, Zeta Test is not designed to function as a tool for performing automated Unit Tests. Instead, Zeta Tests helps you to manually create test cases and then perform and document those test cases manually.

1.1 About this manual

This manual contains an introduction to the Zeta Test application and the most important functions, as well as different usage scenario examples about when and how to use Zeta Test in your projects.

General notes and information is displayed in gray boxes:

Please note:
Information text…

References to the literature index are written in square brackets:

[1]

The optimal way in which to use this manual is to work with Zeta Test running on your workstation; this will allow you to retrace the steps on your own.

1.2 Audience

This manual is written for users who want to learn how to use Zeta Test. Besides that, administrators will learn how to install and maintain Zeta Test in their environment.

When reading this manual, a basic knowledge of and interest in (software) testing, as well as the fundamentals of how to test, will be of great use.

1.3 Copyright/contact

Zeta Test is a product of

zeta software GmbH
Manfred-Wörner-Straße 115
73037 Göppingen
Germany

Internet: www.zeta-test.com
E-mail: info@zeta-software.de

Phone: +49 (7161) 98897-0
Fax: +49 (7161) 98897-29

1.4 Support

zeta software GmbH, the vendor of Zeta Test, is particularly strong in terms of offering fast, in-time support for all its applications.

Accordingly, you will get professional, fast and qualified support from friendly support engineers.

They will be able to give you the best available support, but please use the contact options that are specified on the Zeta Test website [1] only.

2 The idea behind Zeta Test

The idea for Zeta Test was born out of our own needs as a software vendor (ISV, see glossary chapter 5.14). We needed to efficiently test our own applications – both in development and after release. These included not only consumer products for large audiences, but individually developed in-house solutions for business customers too.

Unfortunately we found no existing test applications that fitted our needs; either they had far too many functions we didn't need and were much too expensive, or they had functions which we had no use for.

To summarize, our goal was to produce testing software with which we could ensure that our applications behave exactly how we and the customer expects them to behave. If the tests are performed for changes to existing products, they are called "Regression tests" [2].

We aimed to enable anyone with a manual in their hands to perform tests, and not just the developers of the software themselves.

Very early on, we decided not to get down to the source code level ("Unit testing", [3]) but rather to cover all other aspects of the application like the installation process, the documentation, the usability of the user interface and the behavior of an application on different operation systems.

These were the ideas that guided us in developing Zeta Test.

During the development stage we were pleasantly surprised by the fact that nearly every customer we talked to about the project showed interest in Zeta Test. Due to close partnerships with selected customers from different industries, Zeta Test evolved to an application to cover a broad range of scenarios that goes way beyond our initial intention of simply testing software applications.

2.1 Overview

2.1.1 Scenarios

The workflow when creating, performing and evaluating test cases with Zeta Test is easy to understand and apply, but still comprehensive enough to cover very different application scenarios:

- Internal in-house tests on software created and updated by the software vendor (ISV) itself.
- External tests by companies introducing new software into their operational environment or implementing a new version ("Release management") of existing software within their enterprise.
- Arbitrary tests that have to be planned, performed and then be evaluated.

2.1.2 Test workflow

A typical workflow during testing is as follows:

1. An administrator creates a new *project* with Zeta Test.

2. A project manager creates global *test units* and, within these units, global *test cases*. Test units can be nested arbitrarily. The project manager also has the option of creating *test variants*.

3. A test manager defines *test plans* that will be performed by testers. The test manager adds test units and test cases to the respective test plan. He or she may also group the test plans in *test folders* that can be nested arbitrarily. In addition, the test manager sets permissions for the testers so that only they can access the test plans that are intended for their use.

4. A tester performs the *test runs* he or she has permissions for and works through the individual test cases. Testers execute the tests described and document the *test results* as written text, as well as setting a test result status indicator to show the success of a *completed test case*. There is also the option to, if an external bug tracker database is connected automatically transfer erroneous test results to the bug tracker, thus enabling the developer responsible to correct the error.

5. The test manager and the project manager have cumulated status indicators (traffic lights) in order to have an up-to-date overview of a test run or a test plan. In addition, detailed reports can be shown and exported to common formats (Microsoft Office Excel, Microsoft Office Word, and Adobe PDF).

6. The steps 4 and 6 are repeated as often as necessary until the test manager or project manager is satisfied with the results.

3 Function/scope

Among others, Zeta Test offers the following functions:

- Clean, intuitive, easy-to-use user interface.
- Multi-user environment within your local network (LAN).
- Compatibility with Citrix servers and Microsoft Terminal servers.
- English and German user interface.
- Support for an unlimited number of projects, all being XCOPY deployable.
- User management in stand-alone, Active Directory or mixed mode.
- Comprehensive permission rules.
- Plug-in concept for extending the application.
- Reporting to show and evaluate tests. Export to Microsoft Office Excel or Adobe PDF.
- Connectability to external bug tracker databases like Mantis Bug Tracker.

3.1 System requirements

Zeta Test is a desktop application for Microsoft Windows. The following system requirements must be fulfilled:

- Microsoft Windows XP, Microsoft Windows Server 2003, Microsoft Vista or Microsoft Windows Server 2008.
- Microsoft .NET Framework 2.0 with newest Service Pack. Since it contains Version 2.0, Version 3.5 is also sufficient.
- Zeta Test ships with "VistaDB" [5] as the internal database system. If, however, you plan to use a Microsoft SQL Server database, you need to have Microsoft SQL Server 2005 or 2008 installed, in either the Express, Workgroup, Standard or Enterprise version.

Zeta Test is available with English and German user interface.

3.2 Bug tracker databases supported

Zeta Test currently supports the following bug trackers:

- Mantis Bug Tracker. en.wikipedia.org/wiki/Mantis_Bug_Tracker
- BugZilla. en.wikipedia.org/wiki/Bugzilla
- OnTime. www.axosoft.com/products/ontime.aspx

The connection to a bug tracker runs through the web service interface (SOAP) of the respective bug tracker application.

4 Installing Zeta Test

Zeta Test is installed by running the setup package. A wizard guides you through the installation process.

The setup package comes in two versions: one package is for local installation on one workstation, and one package for the central installation within a local area network (LAN).

4.1 Single-user installation on one workstation

To install Zeta Test on one workstation, download the single-user installation setup package and run it with administrative permissions. Further steps are not required.

All application program files will be installed locally; user settings are also installed locally. You can, of course, use the locally installed version of Zeta Test to access shared projects on your local area network (LAN), too.

4.2 Multi-user installation on a local area network (LAN)

The multi-user installation package enables you to deploy Zeta Test on a central network, allowing all users to use that central installation. All application program files are installed centrally; user settings are also stored centrally. Download and use the network installation setup package and run it with administrative permissions.

You should use the network setup package if you want to collaborate on a Zeta Test project by accessing it from multiple workstations simultaneously.

4.2.1 Central application storage location

In order to keep maintenance costs low, the multi-user installation places all files in a central storage location.

You should therefore create a folder for Zeta Test on your central file server and set permissions for all users who will need to access the folder share.

When you then run the network installation setup package, just specify the central Zeta Test folder on your server as the installation target folder.

The setup wizard copies all the required files into that folder.

After the wizard has finished, you will need to run the *Client Setup* application on each workstation that requires getting access to Zeta Test. The Client Setup is located in the sub folder "ClientSetup\Setup.exe" of the central Zeta Test installation. The process of running the Client Setup is scriptable by using Windows login scripts or Windows group policies.

The advantage of having a central multi-user installation is that you only have to update a single place, i.e. your central Zeta Test file server share, when updating the Zeta Test application. Each workstation then automatically gets the new version when the application is opened.

4.2.1.1 System requirements

To successfully access the central Zeta Test files from a workstation, the following requirements must be fulfilled:

- Each workstation requires .NET Framework 2.0 SP1 or newer to be installed. The easiest way to do this is to use Windows Update or include it in the standard software deployment mechanism of your business.
- The path to the network share where Zeta Test is installed must be located in the "Local Intranet" security zone of each client. You can configure this by using Internet Explorer's security settings or by configuring the workstation's/user's Windows group policies.
- The .NET code access security must have full access to the network path where Zeta Test is installed. The easiest way to set the permissions is to use the included Client Setup.

5 Terms/glossary

In order to make using Zeta Test for the first time as simple as possible, the following chapter shows you an overview of the terms that are used in our software.

5.1 Project/Test project

A **Project** (also called **Test project**) is a complete unit that resides completely within one single folder in the file system (all documents and attachments, as well as the database file itself).

Zeta Test uses the file-based database system "VistaDB" as the default multi-user database. In addition, it supports Microsoft SQL Server 2005 or newer as the database system for a project.

In the latter case, only the documents and attachments are stored in the project folder. The Microsoft SQL Server database is located where your database admin stores the database on the SQL server.

5.2 Test unit

A **Test unit** exists in order to group test cases. When operating with a large number of test cases, it is advisable to group test cases together, e.g. place all logically related test cases into one group.

Test units can be nested as and where needed and each test unit can contain an unlimited number of test cases. Test units are, if you will, similar to folders inside Windows Explorer which group files (for Zeta Test: test cases) together.

Test units are created "globally", i.e. all test units created are independent of any specific test plan. You assign test units and test cases when selecting them for a specific test plan.

5.3 Test case

A **Test case** is a specific description of a single test to perform. A test plan usually contains many test cases that are carried out one after each other (or in a random order) by a tester.

A test case is always contained within a test unit.

Test cases are created "globally", i.e. all test units created are independent of any specific test plan. You assign test units and test cases when selecting them for a specific test plan.

Besides a description of the actions to perform, test cases can also contain an attachment, e.g. example documents to show the expected output of an application being tested.

5.4 Test variant

Test variants are optional elements to further separate test cases. A test variant is always a child element of a test case.

You should use test variants if you have several similar tests cases that share lot of similar properties and only differ slightly from each other.

A test variant inherits all settings from a *base variant* but can override these settings if desired.

5.5 Test folder

A **Test folder** helps you to group test plans. Various test plans are often very different from each other in terms of functionality (e.g. "Test plans for setup", "Test plans for UI tests"). By using test folders, you can group related test plans together.

Test folders can be nested as and where needed and contain an infinite number of test plans. Test folders are, if you will, similar to folders inside Windows Explorer which group files (for Zeta Test: test plans) together.

A test folder has a single cumulated test result (traffic light colors) that is calculated from its child elements (test plans and test folders). This gives you a very quick overview of the overall status of a single test folder.

5.6 Test plan

A **Test plan** is a concrete description of test cases to perform, as well as their results. Test plans can be grouped inside test folders, although this is usually unnecessary for small projects, where you can add them directly to the root.

A test plan contains:

- *A template.*
  This template defines which test units and test cases actually should be tested within the test plan.
- *Test runs.*
  The test runs contain the actual tests that were performed by the tester, as well as their results.

Test runs within a test plan can, if needs be, be grouped by environment (E.g. "Test environment", "Live environment").

A test plan has a single cumulated test result (traffic light colors) that is calculated from its child elements (test environments). This gives you a very quick overview of the overall status of a single test plan.

5.7 Template/test template

A **Template** (also called **Test template**) defines which test units (*planned test units*) and test cases (*planned test cases*) are actually being tested within a test plan.

Usually, you do not test all test units and test cases within one single test plan but instead group them logically into several different test plans. Each test plan therefore contains its own template which defines the actual test units and test cases to test.

You can add test units and test cases to a test plan as long as there are no test runs within the test plan.

The nesting hierarchy of the global test units and test cases can be the same as within a template, but need not necessarily be: i.e. you can change the nesting hierarchy of test units and test cases within a template to be completely different from the original nesting hierarchy within the global test units and test cases section.

In addition, you can add the same global test unit or global test case multiple times to a single template.

5.8 Planned test unit

A **Planned test unit** is a *copy* of a global test unit which is assigned to the template of a test plan.

By creating a planned test unit from a global test unit, you specify that the test unit has to be tested for the assigned test plan (i.e. the test cases contained within the test unit).

Creating a copy of the original test unit (as opposed to just a linked reference to it), you ensure that future changes to the global test unit do not modify the test plan with the planned test unit. This avoids erroneous falsifications of tests that have already been performed.

5.9 Planned test case

A **Planned test case** is a *copy* of a global test case which is assigned to the template of a test plan.

By creating a planned test case from the global test case, you specify that the test case has to be tested for the assigned test plan.

Creating a copy of the original test case (as opposed to just a linked reference to it), you ensure that future changes to the global test case do not modify the test plan with the planned test case. This avoids erroneous falsifications of tests that have already been performed.

5.10 Environment

An **Environment** groups test runs within a test plan. Example values for environments include "Test environment/test system" and "Live environment/live system".

The global project settings allow you to define an unlimited number of environments. For each test plan, you can have as many test runs per test plan as necessary.

An environment has a single cumulated test result (traffic light colors) that is calculated from its child elements (test runs). This gives you a very quick overview of the overall status of a single test environment.

Please note that the cumulated test result of the *most recent* test run finished counts as the test result of the environment.

5.11 Test run

A **Test run** defines the steps that have been carried out within a test plan by a certain test user. A test run is grouped within a test environment.

Test runs can be started by testing users themselves selecting a test plan they have permissions for and then selecting the "Start new test run" command.

The test run contains all test units and test cases that are assigned to the test plan.

The testing user works off the planned test cases (step-by-step or in random order). Each step carried out will be automatically stored as a *performed test case* together with the text that the testing user entered (and optional attachments), together with a test result.

Testing users are able to interrupt a test run at any time and continue it later. In addition, other testing users with the appropriate permissions are able to continue the test run for their colleagues. This is useful if a testing user becomes ill or cannot continue and finish the test run for other reasons.

A test run can be closed if a test result exists for each planned test case.

The sum of the test results (traffic light colors) of all test cases performed will be used as the single cumulated result of the test run.

Please note:
Only the cumulated test results (traffic light colors) of the *most recent* test run *finished* will be propagated as the single cumulated test result of the test run.

5.12 Completed test case

A **Completed test case** is the result of a planned test case within a test run. A completed test case can contain the following information:

- Text with the results of the planned test case.
- Attachments.
- A test result (traffic light colors).

- Information about associated bug tracker database entries (automatically created), provided that the correct bug tracker plug-in is present and configured.

The test results (traffic light colors) of all completed test cases of a test run are cumulated as the total test result (traffic light colors) of the test run.

5.13 Test result

A **Test result** is the result of a completed test case performed within a test run. Test results can be freely defined within a test project (e.g. "Success", "Acceptable", "Failed") and are shown cumulated as traffic light colors.

By configuring the appropriate settings within each test result, you can configure interaction with bug tracker databases.

5.14 Further terms

- **Black-box testing**
  Black-box testing takes an external perspective of the test object as a basis on which to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.
  See also en.wikipedia.org/wiki/Black-box_testing

- **White-box testing**
  White-box testing (a.k.a. *clear-box testing*, *glass-box testing*, *transparent-box testing* and *translucent-box testing* or *structural testing*) uses an internal perspective of the system to design test cases based on internal structure.
  See also en.wikipedia.org/wiki/White-box_testing

- **Regression testing**
  Regression testing is any type of software testing which seeks to uncover software regressions. Such regressions occur whenever software functionality that was previously working correctly stops working as intended. Typically, regressions occur as an unintended consequence of program changes.
  See also en.wikipedia.org/wiki/Regression_testing

- **Change management**
  The Change Management process in Systems Engineering is the process of requesting, determining attainability, planning, implementing and evaluation of changes to a system. It has two main goals: supporting the processing of changes and enabling traceability of changes.
  See also en.wikipedia.org/wiki/Change_management_(engineering)

- **Unit testing**
  Unit testing is a software design and development method where the programmer makes sure that individual units of source code are fit for use. A unit is the smallest testable part of an application.
  See also en.wikipedia.org/wiki/Unit_testing

- **Independent software vendor**
  Independent software vendor (ISV) is a business term for companies specializing in making or selling software, designed for mass marketing or for niche markets.
  See also en.wikipedia.org/wiki/Independent_software_vendor

6 Licensing

Please note:
This chapter contains a brief summary of the licensing terms. For details please visit the Zeta Test website [1] or get in contact with us directly.

6.1 Free license

If you want to use Zeta Test for evaluation purposes, you can do so for free without paying any money or obtaining a license.

In addition, the usage of Zeta Test is completely free-of-charge in the following cases:

- For individual users that use Zeta Test for non-commercial, personal use only.
- For testing Open Source Software (OSS) e.g. if you are writing OSS or if you are planning to introduce an OSS application within your company.
- For use within educational institutions like schools or universities.

6.2 Purchasable license

If you are using Zeta Test for commercial purposes, you will need to acquire the appropriate number of user licenses.

Commercial purposes include – but are not limited – to the following cases:

- For testing commercial software, e.g. if you are developing a non-OSS application that is being sold to customers later.
- For use within your company, e.g. if you are planning to introduce software within your company and want to test this software with Zeta Test.

6.2.1 Concurrent user license model

User licenses are always "Concurrent user" [8]. I.e. the number of licenses that you have to purchase depends on the number of users that are accessing Zeta Test *at the same time*, which is usually different from the total number of users that are accessing Zeta Test.

E.g. if your company has a total of 10 employees that are using Zeta Test but only 5 employees are accessing Zeta Test *at the same time* (i.e. *concurrently*) you have to purchase a 5 user license.

7 Usage of the Zeta Test application

This chapter shows you important elements of Zeta Test and outlines their usage and the idea behind each of them by means of examples.

You find real-world examples on how to use Zeta Test in chapter 8.

7.1 The project structure

All information is stored inside a project ("Test project"). A project is a complete unit that completely resides within one single folder in the file system (all documents and attachments, as well as the database file itself).

You can create an infinite number of projects, each with as many test plans, test cases, etc. as needed.

Depending on your personal preferences, you can either put the test plans of multiple products to test into one project or use a separate project for each one.

Please note:
We recommend that, for multiple, non-related products, you put each product into a separate Zeta Test project. If a project gets too large, you run the risk of losing track of the project as a whole, which could lead to negative effects on performance.

You can create and open an infinite number of projects from within Zeta Test.

In addition, Zeta Test enables you to import test cases from one test project into another test project (see chapter 7.3.2). This is useful when you want to merge or split projects.

If you want to edit multiple Zeta Test projects simultaneously, start a new Zeta Test instance for each project.

7.2 The main window

The main window of Zeta Test is separated into two areas: The *structure view* and the *content area*.

7.2.1 Structure view

The structure view is located on the left side of the main window, being rendered as a tree that lists all elements of a test project.

Please note:
Depending on the filter and project settings as well as the permissions of the user logged in, some elements may not show up in the structure view.

The structure view lists all elements of a project as well the appropriate editing options.

To show the editing options of an element in the structure view, simply right-click on that element. Depending on the type of element, different entries are listed in the context menu which appears.

7.2.2 Content area

The content area consists of multiple tabs (depending on the context) and is shown in the right area of the main window.

The content area displays different content depending on the element selected in the structure view,.

Usually, the content area has an "**Overview**" tab that shows general information about the element currently selected in the structure view. Depending on the context there is also a "**Details**" tab with further details as well as a "**Tabular**" tab that displays cumulated data in a table format.

7.2.2.1 "Overview" tab

The tab "**Overview**" contains the most important information of the element currently selected in the structure view.

7.2.2.2 "Tabular" tab

The tab "**Tabular**" displays a cumulated summary of test results and enables project managers to get a quick overview of the state of a project regarding quality and progress.

The content of the tabular table is exportable as a Microsoft Office Excel document and enables you to further evaluate the results. There is *no need* to have an installed Microsoft Office Excel on your computer for the export to work.

7.3 Working with test cases

Test cases (see chapter 5.3) grouped within test units (see chapter 5.2) define the tests that are available within a project.

A test case describes the prerequisites that must be fulfilled in order to perform the test, the test steps that must be performed during testing and the expected results of the completed tests.

A test case can contain an infinite number of attachments, e.g. a Microsoft Office Excel document that describes how the software to be tested should behave (i.e. in this example, that the application to test generates Microsoft Office Excel documents by itself).

Please note:
The dialog window for editing test cases is non-modal, i.e. you can keep the dialog window open and still continue working with other parts of Zeta Test. This is helpful when you e.g. want to display multiple information concurrently.

7.3.1 Creating and editing test cases

In order to create a test case, follow these steps:

1. Select a test unit in the structure view.
2. Right-click the test unit.
3. Select "**New test case**" from the context menu.
4. The dialog window "**Edit test case**" shows up. Fill out all required fields.
5. Confirm by clicking "**OK**".

Follow the following similar steps to edit an existing test case:

1. Right click the test case to edit.
2. Select "**Edit test case**" from the context menu.
3. The dialog window "**Edit test case**" shows up. Fill out all required fields.
4. Confirm by clicking "**OK**".

Please note:
If planned test cases exist that are assigned to a test case (within the respective test plans), the changes to a test case do *not* automatically change the planned test cases, too. You will need to perform this action manually. To do this, open the "**Edit test plan**" dialog window, select the "**More settings**" tab and click on the "**Synchronize test case modifications**" button.

7.3.2 Importing test cases

You can import test cases from other projects:

1. Right click "Test units" in the structure view of the main window.
2. Select "**Import test units and cases**" from the context menu.
3. Follow the instructions in the wizard dialog that appears.

Test cases that you imported into a project still exist in the original project from which they were imported without any changes to them.

7.4 Working with test variants

Test variants (see chapter 5.4) are optional elements that allow you to further subdivide test cases.

7.4.1 Creating test variants

In order to use test variants, follow these steps:

1. Select menu item "**Test variants**" from the "**Project**" main menu item.
2. Select the "**Settings**" tab.
3. Ensure that the "**Use test variants**" option is set to "Yes".

Then define the test variants:

1. Select menu item "**Test variants**" from the "**Project**" main menu item.
2. Select the "**Test variants**" tab.
3. Use the "**New**" button to create a new test variant.

A typical test variant would have e.g. the name "Operating System" and contain the values:

- "Windows XP"
- Windows Vista"
- "Mac OS X"

The variant, as an example, therefore defines the different operating systemson which to test the application.

Another test variant could e.g. be named "Language" and define the different languages that are important to test, e.g.:

- "English"
- "German"
- "French"
- "Italian"

Later, when assigning test variants to test cases, you will be able to add test variants - or combinations of test variants - to a test case.

Please note:
The ability to combine zero or more variants together and add it to a test case results in a very flexible system that can be used in a wide range of scenarios.

7.4.2 Creating test variant presets

Using presets ("predefined combinations of variants") is an *optional* comfort-function that enables you to predefine frequently-used variant combinations and easily add them to test cases later on.

By using presets you increase the rate at which you work and reduce the number of errors resulting from incorrect assigned test cases.

That said, please note that using presets is completely optional.

To define presets, follow these steps:

1. Select menu item "**Test variants**" from the "**Project**" main menu item.
2. Select the "**Presets**" tab.
3. Use the "**New**" button to create a new preset.

You can then define your desired combinations In the "**Edit test variant preset**" dialog window that appears,.

Please note:
Use the "**Default test variant**" field on the "**Settings**" tab in the global "**Test variants**" dialog window to define presets that are automatically used when creating new test cases.

7.4.3 Assigning test variants to test cases

To actually use the previously defined test variants, you have to assign them to the desired test cases.

To do this, you will need to start by creating a test case as described in chapter 7.3.1.

To assign variants follow these steps:

1. Open the "**Edit test case**" dialog window and click on the "▼" button in the "**Variant**" field.
   → A context menu will open.
2. Either use the "**Apply preset**" sub menu to directly apply the variants of a test case, or use the menu item "**Edit list**" to define test variant combinations for the test case currently being edited.

7.5 Planning of test runs

Use test plans (see chapter 5.6) to create the templates for the test runs that are performed later by the testing users.

You should create the test plans so that the testing users can work through related areas together in one test run: e.g. if you are testing a software project, you could create different test plans for testing setup installation packages on different platforms, or you could create one test plan for each platform that covers all the different areas across one program version.

To create a test plan, follow these steps:

1. In the main window, select the parent element in the structure view. This is either the "Tests" node or the respective test folder.
2. Right-click the element.
3. Select "**New test plan**" from the context menu.
   → The "**Edit test plan**" dialog window will open.
4. Enter you desired values. Usually, you will only need to fill in the "**Title**" field.
5. Confirm with "**OK**".

The test plan will then be created and will be shown in the main window of the structure view. Add the test cases and test units that you want to be tested to the test plan.

Follow these steps to do so:

1. Select the test plan in the structure view of the main window.
2. Right-click the test plan.

3. Select "**Add test units and cases**" from the context menu.
   → The dialog window "**Select test units and test cases**" will open.
4. Select the test units and test cases that you want to be tests.
5. If activated, you can now select the test variants to test.
6. Confirm with "**OK**".

After this, the test units and test cases will be added to the test plan.

Please note:
You can add a test case multiple times and at a hierarchy different to that originally intended by the author of the test case. This can be done by selecting the desired parent element below the "Template" element of a test plan and by selecting "**Add test units and cases**" from the context menu that opens.

After this step, a testing user can run test runs for the test plan you created (see chapter 7.6).

Please note:
You can add test units and test cases for test plans that have no test runs yet.

7.6 Performing test runs

After you have filled a test plan with all desired test units and test cases, one or multiple testing users can perform test runs (see chapter 5.11).

To start a new test run, follow these steps:

1. Select the test plan in the structure view of the main window.
2. Right-click the test plan.
3. Select "**Start new test run**" from the context menu.
   → The dialog window "Start test run" opens.
4. Select the elements to test.
5. Confirm with "**OK**".
   → The test run will be created.
   → The dialog window "**Perform tests**" will open.
6. Perform the tests and enter the results.
7. Click on the "**Next**" button to navigate to the next test.
8. If you have performed all tests or want to pause the test prematurely, click on the "**Close**" button.

You can continue a test you have paused at any time.

To continue a non-closed test run, follow these steps:

1. Select the test run in the structure view of the main window.
2. Right click the test run.

3. Select "**Continue tests**" from the context menu.
   → The dialog window "**Perform tests**" will open.
4. Perform the tests and enter the results.
5. Click on the "**Next**" button to navigate to the next test.
6. If you have performed all tests or want to pause the test prematurely, click on the "**Close**" button.

Please note:
The dialog window for performing tests is non-modal, i.e. you can keep the dialog window open and still continue working with other parts of Zeta Test. This is helpful when you e.g. want to display multiple information concurrently.

7.7 Grouping test plans

Using test folders (see chapter 5.5) gives you an efficient tool to logically group test plans in large projects. In addition, test folders help you to assign user permissions in an efficient manner.

Please note:
In order to work with test folders you must activate the use of test folders in the project settings (see chapter 7.11.1).

Just like normal folders in your Windows file system, you can nest test folders as and when required: i.e. a test folder can contain an infinite number of child test folders, as well as an infinite number of test plans.

To create a test folder, follow these steps:

1. Select the parent element in the structure view of the main window.
2. Right-click the element.
3. Select "**New test folder**" from the context menu.
   → The dialog window "**Edit test folder**" will open.
4. Enter the desired information. Usually, you will only have to fill in "**Title**" field.
5. Confirm with "**OK**".

7.8 Attributes

Attributes are name value pairs that can be attached to different elements within the hierarchy of a project (e.g. to test cases). Attributes can also be inherited from parent hierarchy elements to child hierarchy elements.

Attributes are therefore a general purpose functionality of Zeta Test to add additional information (in the form of name value pairs) to existing elements.

Please note:
Attributes are a completely *optional* feature. For most of your Zeta Test projects, it is very likely that you never need to use attributes at all.

Once attached to an element the attributes are, among others, accessible in the following contexts:

- In reports (see chapter 7.9).
- In the tabular content area of the main window (see chapter 7.2.2).
- In the "**Overview**" tab in the content area of the main window (see chapter 7.2.2).

7.8.1 Attribute types

Besides a name and a value, attributes can also be given a type. The type controls how the user enters values for an attribute and how values entered are then displayed.

The types available are:

- Text box, single line
- Text box, multi line
- Date selection
- Check box
- List

7.8.2 Attribute styles

Attribute styles are an advanced feature with which you can define a template of "attributes to create" that can be used later in the workflow.

An attribute style contains one or multiple style fields that define the name and type of the attributes that are being created based on that attribute style.

7.8.2.1 Defining and assigning attribute styles

To create, edit or delete attribute styles, follow these steps:

1. Click on menu item "**Attribute styles**" in the "**Project**" main menu item.
   → The "**Attribute styles**" dialog window will open.
2. On the "**Attribute styles**" tab, use the button "**New**", "**Edit**" or "**Delete**".

You also can define standard attribute style for the various element *types* in the structure view of the main window. When being defined, these styles are available in the property dialog windows of the elements on the "**Attribute**" tab, in the context menu of the "**New**" button.

To define standard attribute styles follow these steps:

1. Click on menu item "**Attribute styles**" in the "**Project**" main menu item.
   → The "**Attribute styles**" dialog window opens.
2. Click on the "**More settings**" tab.

3. For each element listed, select the standard attribute style.

7.8.3 Promoted attributes

There is another input method for attributes, used for entering attributes for test cases, as well as during a test run.

In these scenarios, attributes cannot only be created on the "**Attributes**" tab, but also listed side-by-side on the very first tab of the dialog window. This simplifies entering of attributes for the users.

To configure an attribute to be enterable on the first tab page, you have to mark the respective attribute style as being a "promoted attribute".

To do so, follow these steps:

1. Click on menu item "**Attribute styles**" in the "**Project**" main menu item.
   → The "**Attribute styles**" dialog window will open.
2. On the "**Attribute styles**" tab, use the button "**New**" to create a new attribute style, or use the button "**Edit**" to edit an existing attribute style.
   → The "**Edit attribute style**" dialog window will open.
3. Use the "New" button to add an attribute style field.
   → The "**Edit attribute style field**" dialog opens.
4. Set the value of the field "**Is promoted**" to "Yes" in the "**Settings**" field.
5. Close all dialog windows with "**OK**".

After these steps, it is important that you assign the respective attribute style as a standard attribute style for *test cases* or *test run result items*. Otherwise Zeta Test cannot detect which attribute style to use for displaying promoted attributes.

Please note:
Be careful when configuring promoted attributes to ensure that you changes do not erroneously change already finished test runs.

7.9 Working with reports

Reports help you to extract, prepare and export information. Reports are usually PDF documents that are dynamically generated from Microsoft Office Word documents.

To generate reports, follow these steps:

1. Select the element (e.g. a test plan or a test run) from the structure view of the main window.
2. Right-click the element.
3. Select the sub menu "**Reports**" from the context menu.
4. Depending on the element, you will find zero or more reports in the sub menu.
5. Click on a report to let Zeta Test generate and display the report.

Please note:
Not all elements in the structure view of the main window provide reports.

7.9.1 Internal workflow during report generation

PDF reports are generated dynamically by a flexible mechanism:

1. A Microsoft Office Word document with mail merge placeholder fields is loaded for a report.
2. Zeta Test dynamically builds an internal application context with a set of mail merge placeholder fields and values (see chapter 7.9.2.1).
3. The mail merge fields in the loaded Microsoft Office Word document are replaced and expanded.
4. The Microsoft Office Word document is then converted internally into a PDF document.
5. The PDF document is displayed.

Please note:
You need *no* Microsoft Office and *no* Adobe Acrobat installed on your computer in order to generate the reports; Zeta Test already ships with all the functions.

7.9.2 Creating and customizing reports

You can customize the report generation process by personalizing the Microsoft Office Word documents we ship or by creating an infinite number of completely new Microsoft Office Word documents by yourself.

The storage location of the Microsoft Office Word documents is relative to the "**Resources**" folder of the Zeta Test installation in the sub folder of the language you have set ("**ENU**" or "**DEU**"). There, it is in a folder with the name of the element type in the structure view in the main window: e.g. the sub folder "**TestPlan**" contains the reports for test plans.

In addition to the global storing of reports, you also can store reports on a per-project basis, relative to the base folder of the project in the same manner as described above.

In order to create a new Microsoft Office Word document for a new report, create a Microsoft Office Word document in the ".doc" format (".docx" is currently not supported) and store it in the desired folder.

Use the following naming scheme: "report-<*NameInContextMenu*>.doc": i.e. "<*NameInContextMenu*>" is the name that will be shown later in the "**Reports**" context sub-menu in Zeta Test: e.g. if you would call your report "report-My first report.doc", the respective element in the context menu would be called "My first report".

Please note:
The best way to understand how to create your own reports is to take existing reports, copy them and customize them.

The next chapter shows you how to access the available mail merge fields in your own reports.

7.9.2.1 Reference of available mail merge fields

An overview of all available Microsoft Office Word mail merge fields for creating reports is directly available within Zeta Test:

1. Click on the "**?**" main menu, then click on the "**Diagnostic center**" menu item.
2. Click on the "**Hints**" tab.

7.10 User management

Zeta Test is designed for multiple users to work on the same project simultaneously over local area networks (LAN). The user management controls which users may perform which actions.

To open the user management, follow these steps:

1. Click on the "**User management**" menu item in the "**Project**" main menu.
   → The "**Manage groups and users**" dialog window opens.
2. Use the tabs "**Users**", "**User groups**" and "**Active Directory**" to manage the different areas.

The user management's scope is always on a per-project-basis.

7.10.1 User groups

User groups can contain one or multiple users (members). Every user group has an assigned role.

A user can be member of more than one user groups. Therefore, a user can have multiple roles assigned by its multiple group memberships.

Please note:
User groups cannot be nested, i.e. a user group *cannot* be member of another user group.

To edit user groups, use the buttons "**New**", "**Edit**" or "**Delete**" on the "**User groups**" tab.

7.10.1.1 Roles

Each user group in Zeta Test has an assigned role. A role is a sum of permissions that is typically required for performing a certain task.

Zeta Test defines the following roles:

- **Administrator**
  An administrator has the highest set of permissions. He has all the permissions of a project manager and, additionally, can configure all aspects of a test project.

- **Project manager**
  A project manager has all the permissions of a test manager and can also create test units and test cases. He or she also can start and edit test runs as well as assign permissions for test plans.

- **Test manager**
  A test manager can perform most of the project manager tasks, but has only limited permissions to access certain elements.

- **Tester**
  A tester can only perform tests where he or she was granted permissions to.

7.10.2 Users

To edit user, use the buttons "**New**", "**Edit**" or "**Delete**" on the "**Users**" tab.

7.10.2.1 Integrated authentication

You can configure a user to use Zeta Test's integrated authentication method:

1. Open the "**Edit user**" dialog window.
2. Select the "**More options**" tab.
3. Uncheck the check box "**User is authenticated through Active Directory**".
4. Select the "**User**" tab.
5. Enter a log in password in the fields "**Password**" and "**Password repeat**".

Please note:
Password repetition is there to ensure that the password is entered correctly.

7.10.2.2 Authenticating through Active Directory

If you want to configure a user so that he or she can log in with the same password as the Active Directory user account, follow these steps:

1. Open the "**Edit user**" dialog window.
2. Select the "**More options**" tab.
3. Check the check box "**User is authenticated through Active Directory**".
4. Select the "**User**" tab.
5. Enter the domain as well as the Windows log in name of the user in the form "MyDomain\MyUser" in the field "**Login name**".

7.10.3 Importing from Active Directory

Please note:
Active Directory is a directory service from Microsoft and included in Windows Server 2000, Windows Server 2003 and Windows Server 2008. Active Directory is a directory service used to store information about the network resources across a domain and also centralize the network. Importing from Active Directory into Zeta Test should be done by people with appropriate administrative knowledge only.

You can import users and user groups from any number of different Active Directory domains into Zeta Test.

To configure the import of users and user group, use the buttons "**New**", "**Edit**" or "**Delete**" on the "**Active Directory**" tab.

To actually perform an import, select an import configuration from the list "**Import configurations**" and click on the "**Import**" button.

> Please note:
> You can configure Zeta Test to automatically import from Active Directory. See chapter 9.1.2 for details.

7.10.4 Permissions on test folders, test plans and test units

Beside the permissions that are derived from the roles a user (through the user group) has, you can explicitly specify access permissions for user groups on test folders, test plans and test units.

These permissions are not configured in user management, but directly in the properties dialog window of the object, e.g. in the properties dialog window of a test folder.

The following permissions can be assigned:

- **Full access**
  The members of the configured user groups have full access to the element.
- **Read-only access**
  The members of the configured user groups may view the element, but cannot edit it.

Depending on the type of object, you can configure a permission to be inherited by checking the "**Entail**" check box. If the option is checked, the configured permissions apply to the element itself, as well as to all other child elements, both existing and later added.

If a child element has the same permission configured as an inherited permission in a parent element, the permission on the child element overwrites the inherited permission.

7.10.4.1 Assigning permissions

To assign permissions, follow these steps:

1. Select the test plan, test folder or test unit in the structure view in the main window where you want to set permissions for.
2. Right click on the element.
3. Select "**Edit**" from the context menu.
4. Select the "**Permissions**" tab.
5. Change or create permissions by using the button "**New**", "**Edit**" or "**Delete**".

7.11 Project settings

You access the project settings by clicking the "**Settings**" menu item in the "**Project**" menu.

Project settings are directly written into the project database of the project currently loaded, partially also directly into the configuration file of this project (stored in the root folder of the project).

Project settings therefore only apply to one project and are - independent of the user who is at that moment logged in - the same for all users.

Please note:
Please be very careful when configuring project settings. Changes of the project settings can have various impacts; e.g. if you change test results (tab "**Test results**", see chapter 7.11.4) you could accidently change the meaning of complete test runs.

7.11.1 "Settings" tab

Use this tab to configure general project settings.

Entries in the "**Project configuration file**" section are not stored in the database but rather in the project configuration file "ZetaTest.ztproj". The value of the field "Alternative connection string" is set automatically when moving the project to Microsoft SQL Server (see chapter 7.14.3).

Please note:
Usually all values in property grids like the one on the "**Settings**" tab are written in **bold** if they contain non-standard values that have been modified by the user. Non-bold values indicate standard values.

7.11.2 "More settings" tab

On the "**More settings**" tab you configure if and how you want the project to connect to an external bug tracker database system.

If a bug tracker database system connection is configured, testing users can automatically and semi-automatically create entries into the bug tracker database system upon erroneous test results.

Please note:
The bug tracker database system is *not* a component of Zeta Test, but will be purchased separately from the vendor of the bug tracker database system itself and installed and configured by your administrator. He is the one who can tell you the appropriate settings to configure in Zeta Test in order to successfully connect to the bug tracker database system.

7.11.3 "Description" tab

Use this tab to enter a description of the project: you may phrase the description as you like.

The description appears in no other place of your project and is solely intended to allow you to record internal notices: e.g. project-relevant information that is important to remember.

7.11.4 "Test results" tab

Zeta test displays cumulated test results as traffic light colors in the structure view of the main window. Therefore results of child elements are being propagated to parent elements, resulting in each node in the tree showing the complete state of the respective branch.

That said, Zeta Test is aware of the fact that different scenarios exist and must be fulfilled. Therefore, you are able to configure the test results in a very flexible way.

The tab "**Test results**" enables you to create as many test results as you require. These can later be used by users when carrying out tests.

Please note:
Even though you can define an unlimited number of test results, you have to assign one out of three traffic light colors (green, yellow, red) to each defined test result in order for Zeta Test to always be able to correctly cumulate test results.

7.11.5 "User tools" tab

Use this tab to configure new menu items that will appear in the "**Extras**" main menu. This is helpful when you want to launch related external resources from within Zeta Test in a simple and effective manner: e.g. navigating to an URL or launching Windows Explorer.

Optionally use placeholders, see chapter 9.2.

Example 1

Windows Explorer := $(SpecialFolder.Personal)

This opens Windows Explorer and goes directly to the folder with your personal documents.

Example 2

Visit Microsoft website := http://www.microsoft.com

This would open your standard web browser and navigate to the Microsoft website.

7.11.6 "Attachments" tab

Use this tab to store attachments that are "global" for the project.

The attachments are being used at no other place in the project and are intended to provide another option to include information into the project.

7.11.7 "Attributes" tab

Use this tab to add global attributes to the project.

If an element in the structure view in the main window tries to access an attribute, this means that a search higher up the inheritance hierarchy was unsuccessful and that it is now using the global attributes as a last resort.

For details on attributes, see chapter 7.8.

7.11.8 "Information" tab

This tab shows read-only internal information.

The tab is e.g. useful when you want to examine details during resolving of errors.

7.12 Personal project settings

In contrast to the project settings (chapter 7.11) that are valid for all users of a project, the personal project settings are valid for one user only and do not affect other users.

Personal project settings enable a user to configure parts of the application the way he or she wishes.

To change the personal project settings of the currently logged in user, follow these steps:

1. Click on the "**My personal project settings**" menu item in the "**Project**" main menu.
2. Modify the desired settings.
3. Confirm by clicking "**OK**".

To change the personal project settings of a specific user, follow these steps:

1. Click on the "**User management**" menu item in the "**Project**" main menu.
2. Select the user and click on "**Edit**".
3. Select the "**More options**" tab.
4. Click on the "**Project settings**" button.
4. Modify the desired settings.
5. Confirm by clicking "**OK**".

7.13 Application options

The application options differ from the project settings (chapter 7.11) that are valid for all users of one project and from the personal project settings (chapter 7.12) that are valid for one user of one project, inasmuch as they are independent of a project and are valid for all projects.

By default, the application options are stored per workstation and per user so that each user can configure his or her options independently of other users.

By using the application configuration file "zetatest-core.exe.config" in the applications folder, you control where the program options are stored. By using this mechanism you can for example arrange

for a user to always get the same application options, no matter from which workstation he or she logs in.

7.13.1 Defining the storage location

> **Please note:**
> You should perform the steps described here only if you have the appropriate administrative knowledge and have experience in working with XML files.

To define the storage location, follow these steps:

1. Open the application configuration file of Zeta Test named "zetatest-core.exe.config" located in the application's installation folder with a text editor, e.g. the Windows editor "Notepad".
2. Below the XML element "configuration/appSettings" insert the following entry:
   `<add key="alternativeSettingsFolderPath" value="`*`PathToOptionsFolder`*`" />`
3. Restart Zeta Test

"*PathToOptionsFolder*" is the relative or absolute path to the desired folder. Or you can use placeholders (see chapter 9.2). An example would be:
„$(SpecialFolder.ExecutingAssembly)\..\Settings\".

> **Please note:**
> A complete list of supported "appSettings" entries is available from within Zeta Test. Click on the main menu "**Extras**", menu item "**Options**", tab "**Information**", section "**Application configuration file**".

7.14 Further options

7.14.1 Search

The menu item "**Find**" in the "**Edit**" main menu opens the "**Find**" dialog window which allows you to run a full-text search in your project.

You can directly navigate to the element in the main window's structure view by double-clicking it or clicking on it and then clicking on the "**Go to**" button.

> **Please note:**
> The search result my only show you a subset of the actual results, depending on your permission on certain elements.

7.14.2 Integrating external applications

You can add shortcuts to external application in the "**Extras**" menu. These shortcuts are available to all users of a project.

For details, see chapter 7.11.5.

7.14.3 Migrate database to Microsoft SQL Server

By default, Zeta Test uses a VistaDB database to store your project information.

For performance and scalability reasons it may be necessary to use a Microsoft SQL Server database instead of the VistaDB database.

Zeta Test supports you in migrating from VistaDB to Microsoft SQL Server (and vice versa) with a powerful integrated wizard.

To start the wizard, select main menu "**Project**", menu item "**Advanced**", then sub menu item "**Move a project from/to Microsoft SQL Server**".

Please note:
The menu item is only available if no project is currently loaded. Therefore close any open project before.

The wizard starts and guides you through the process of moving. You can migrate in both directions, from VistaDB to Microsoft SQL Server as well as from Microsoft SQL Server to VistaDB.

8 Usage scenarios

To give you some ideas in which scenarios you could use Zeta Test, the following examples contains description of usage scenarios.

8.1 Example: Developing a new version of an out-of-the-box application for creating public websites

Initial conditions:

You are the project manager of a software vendor company developing a content management system (CMS) called "EasyWebsite" that is also being sold through the company's online shop.

The version of EasyWebsite currently on the market is 7.0; the version under development is 8.0.

Requirements:

You want to test version 8.0 and make it available to all new and existing customers afterwards.

Therefore, you have to ensure that both new customers that do not yet own version 7.0 - as well as existing customers that will update from version 7.0 to version 8.0 - will be able to successfully work with the new version 8.0.

Since you have a rather heterogeneous customer base, you will have to take differing operating systems as well as different language and regional settings into account.

Methods to achieve the task:

You create a plan:

- Which target systems have to be tested?
- Which test must be performed?
- How many testing users and test managers do I need?

You can carry out these planning steps directly within Zeta Test by creating test cases and test plans. In addition, you define the required target systems and environments to test for.

After that, you let your administrator set up and configure the appropriate hardware test environment (or indeed as a virtual machine).

The testing users then carry out the tests you defined. With the integrated bug tracking database interface, the developers of EasyWebsite immediately get notified if a test fails. They can then correct the errors reported, probably after discussing the issues with the testing users.

As soon as all errors have been corrected by the developers, a new version of EasyWebsite is deployed into the test environment and all tests are carried out again.

This workflow is repeated until you as the project manager decide (by looking at the reports generated by Zeta Test) that the new version of EasyWebsite is stable and mature enough to ship to your customers.

Benefits:

After successful completion of the tests, you have strong proof that your software will run on your clients computers successfully.

You also get a full documentation of all tests and test results. For upcoming future versions of EasyWebsite, you have a test plan basis that is easy to enhance with the new features.

After all, you have a well-tested application with a minimum of time spent: and your users will be satisfied customers.

8.2 Example: New version of an internal application to record travelling expenses on a local area network (LAN)

Initial conditions:

You are the administrator in a company with 100 employees that sells products through a network of regional sales employees.

You use an internal application "EasyDrive" within your local area network (LAN) to record and bill traveling expenses. EasyDrive currently runs in version 2.3.

Approximately 30 regional sales employees use EasyDrive to record their traveling expenses on a daily/weekly basis. An interface to your existing ERP system enables EasyDrive to pass on the data.

The vendor of EasyDrive recently shipped an update to version 3.0. Due to your current maintenance contract, your company receives the update free of charge.

Requirements:

You want to roll-out the update and ensure that this happens as seamlessly as possible.

Since you want to ensure that the new Version of EasyDrive runs correctly on all installed workstations, you have to create a test environment (as hardware or as virtual machines), describe test cases and perform tests.

The protocols of the tests performed will help you to decide whether EasyDrive runs as expected and can be deployed into a live business environment.

Methods to achieve the task:

You create a plan:

- Which tests have to be performed?
- Who are the testing users and who are the test managers?

You do these planning steps directly within Zeta Test by creating test cases and test plans. Since all employees use the same type of PC hardware and operating system, you define a single test environment that corresponds to the workstation PC of an employee.

After that you let your administrator set up and configure the appropriate hardware test environment (or indeed as a virtual machine).

You receive the new version 3.0 of EasyDrive from the vendor and deploy it into your test environment.

The testing users perform the tests that you defined previously. If errors occur during these tests (i.e. tests are failed), a test manager generates a report with all failed tests at the end of a test run and forwards these errors to the vendor of EasyDrive.

The vendor of EasyDrive corrects the errors, provides you with a new version that you again deploy into your test environment. After that the testing users perform all tests again.

This workflow is repeated until you as the project manager decide (by looking at the reports generated by Zeta Test) that the new version of EasyDrive can be deployed into a live business environment.

Since EasyDrive is a high-availability application, you have to perform all tests in the live production environment again, to ensure that the application behaves as expected there too. Therefore, the testing users create copies of the test plans for the test environment and perform the tests in the live environment.

If tests in the live environment fail, a rollback of the deployment is implemented and the previous version is restored. You will notice the vendor of EasyDrive and the tests will start again in the test environment.

If everything then works correctly in the live environment, the tests are finished. For revision purposes, you can generate reports of the individual test runs at any time.

Benefits:

Due to using Zeta Test, all functions and behaviors of the new applications were extensively tested.

You can support the go-live of the application with a smaller team then you would have done without Zeta Test because the feedback from the users will be less than ever.

Overall, your users will have high levels of satisfaction with the software.

9 Appendix

9.1 Command line arguments

You can call Zeta Test (either "zetatest.exe" or directly "zetatest-core.exe" with command line arguments.

This enables you to e.g. use Zeta Test within automatic batch scripts or to change certain default values when the application starts.

9.1.1 Language of the user interface

You can select the graphical user interface either in **English** or **German**.

Zeta Test automatically selects the language in accordance with the language of your operating system.

If you would like to change the language manually (e.g. if you are working with a German operating system but wish to work with Zeta Test in English), simply enter the following command line parameter:

-language=<*Language abbreviation*>

The following abbreviations, each three letters long, are currently supported:

- ENU: English
- DEU: German

A valid input to start Zeta Test in English would be:

C:\Program Files\Zeta Test\Applications\zetatest.exe" -language=ENU

9.1.2 Importing from Active Directory

Syntax:

-importad <*Import configuration to execute*>

Description:

Imports users and groups with the specified import configuration from the Active Directory (AD).

9.1.3 Overview of all parameters

An overview of all available command line parameters is directly available within Zeta Test:

1. Click on the "**?**" main menu, then click on the "**Diagnostic center**" menu item.
2. Click on the "**Hints**" tab.

9.2 Path placeholders

At various locations within the Zeta Test application - usually when specifying paths for folders and files - you can use placeholders.

An overview of all available placeholders and their descriptions is directly available within Zeta Test:

1. Click on the "**?**" main menu, then click on the "**Diagnostic center**" menu item.
2. Click on the "**Hints**" tab.

9.3 Folder structure

This chapter explains the folder structure of a Zeta Test installation (see also chapter 4).

9.3.1 Single-workstation installation

Taking an English Windows Vista system with a "C:" drive and a user John Doe ("jdoe"), a single user installation will have the following folder structure:

- "**C:\Program Files\Zeta Test**" – Main folder of the installation.
  - "**Applications**" – Sub folder with application files.
  - "**Packaging**" – Sub folder with various resources copied into the local user folder at first application startup.
- "**C:\Users\JDoe\Documents\Zeta Test**" – User folder with write permissions for the user.
  - "**Projects**" – Sub folder with the Zeta Test projects of the user.
  - "**Resources**" – Sub folder with the HTML files that are shown in the content area of the main window (see chapter 7.2.2).
  - "**Settings**" – Sub folder with the application settings.
  - "**Stationary**" – Sub folder with the (multi lingual) templates/files for the first application startup and with the project that serves as the template when creating new projects.

By changing settings in the application configuration file, most of the folder locations can be changed. I.e. the application does not access the folders described above but the alternative folder paths you specified instead.

See chapter 7.13.1 for details on how to change the folder locations.

9.3.2 Network installation on a local area network (LAN)

Currently, there is no installation setup package available for directly installing onto your LAN. Instead, please follow the steps described in chapter 4.2 and create the folder path manually afterwards.

Then, you will need to adjust the application configuration file so that Zeta Test knows the new folder paths. See chapter 7.13.1 for details.

A suitable example folder structure might be as follows:

- "**\\myserver\myshare\Zeta Test**" – Main folder of the installation.
  - "**Applications**" – Sub folder with application files.
  - "**Packaging**" – Sub folder with various resources copied into the local user folder at first application startup.
  - „**Projects**" – Sub folder with the Zeta Test projects for all users.
  - "**Resources**" – Sub folder with the HTML files that are shown in the content area of the main window (see chapter 7.2.2).
  - "**Settings**" – Sub folder with the application settings.
  - "**ClientSetup**" – Sub folder with "setup.exe" that will be executed in a multi-user installation once on each accessing workstation. See chapter 4.2 for details.
  - "**Stationary**" – Sub folder with the (multi lingual) templates/files for the first application startup and with the project that serves as the template when creating new projects.

In the example above *all* folders are stored in one location, independently from the user logged in. It is important in such a case that all users have write permissions for the folders, which are normally stored locally in the user's personal folder (see chapter 9.3.1).

10 Index

U

V

W

X

Y

Z

11 Literature

The following sources were being used for this manual:

1. Zeta Test website. www.zeta-test.com
2. "Regression testing". en.wikipedia.org/wiki/Regression_testing
3. "Unit testing". en.wikipedia.org/wiki/Unit_testing
4. „Test case". en.wikipedia.org/wiki/Test_Case
5. VistaDB database. www.vistadb.net
6. "Black-box testing". en.wikipedia.org/wiki/Black-box_testing
7. "White-box testing". en.wikipedia.org/wiki/White-box_testing
8. "Concurrent user". en.wikipedia.org/wiki/Concurrent_user